

Roll Number:17134

Vishal Kokane

Assignment 002

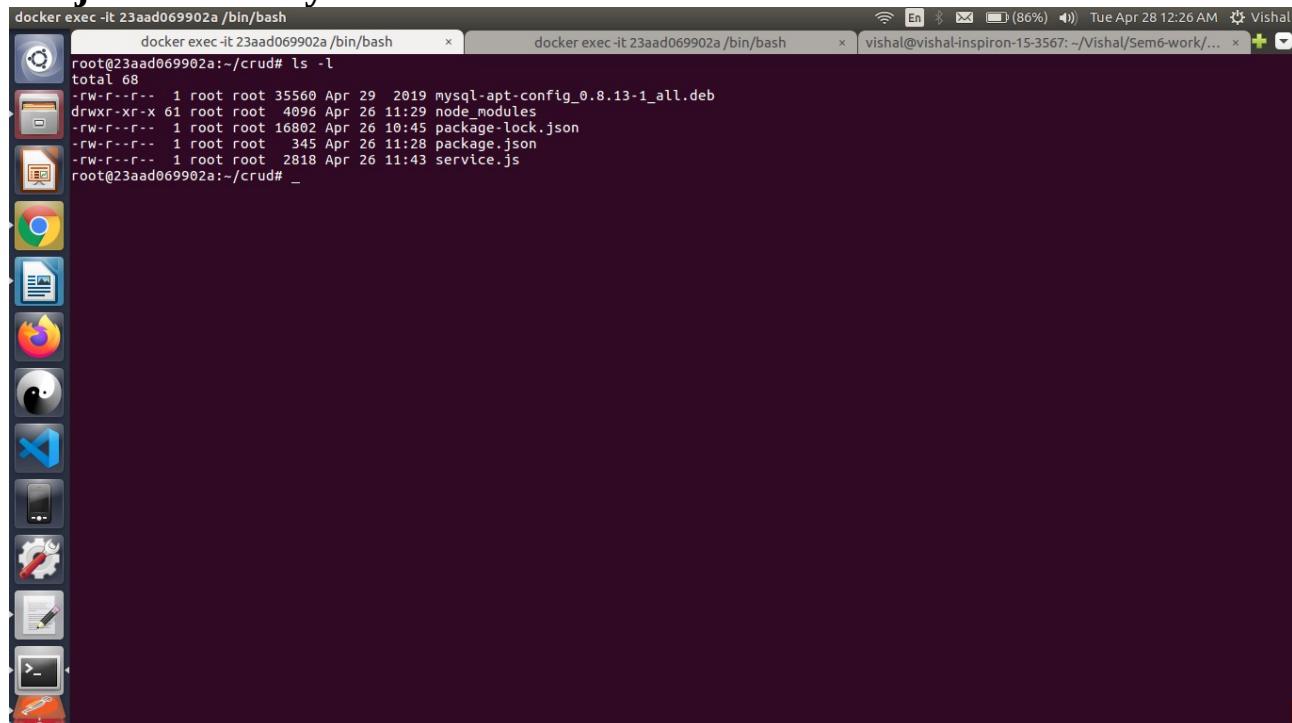
---

This assignment is of building REST api using CRUD operations.

I've used nodejs +mysql in base docker image of ubuntu.

Git repository url:<https://github.com/vishalkokane264/pucsd2020-devops>

### Project Directory structure:



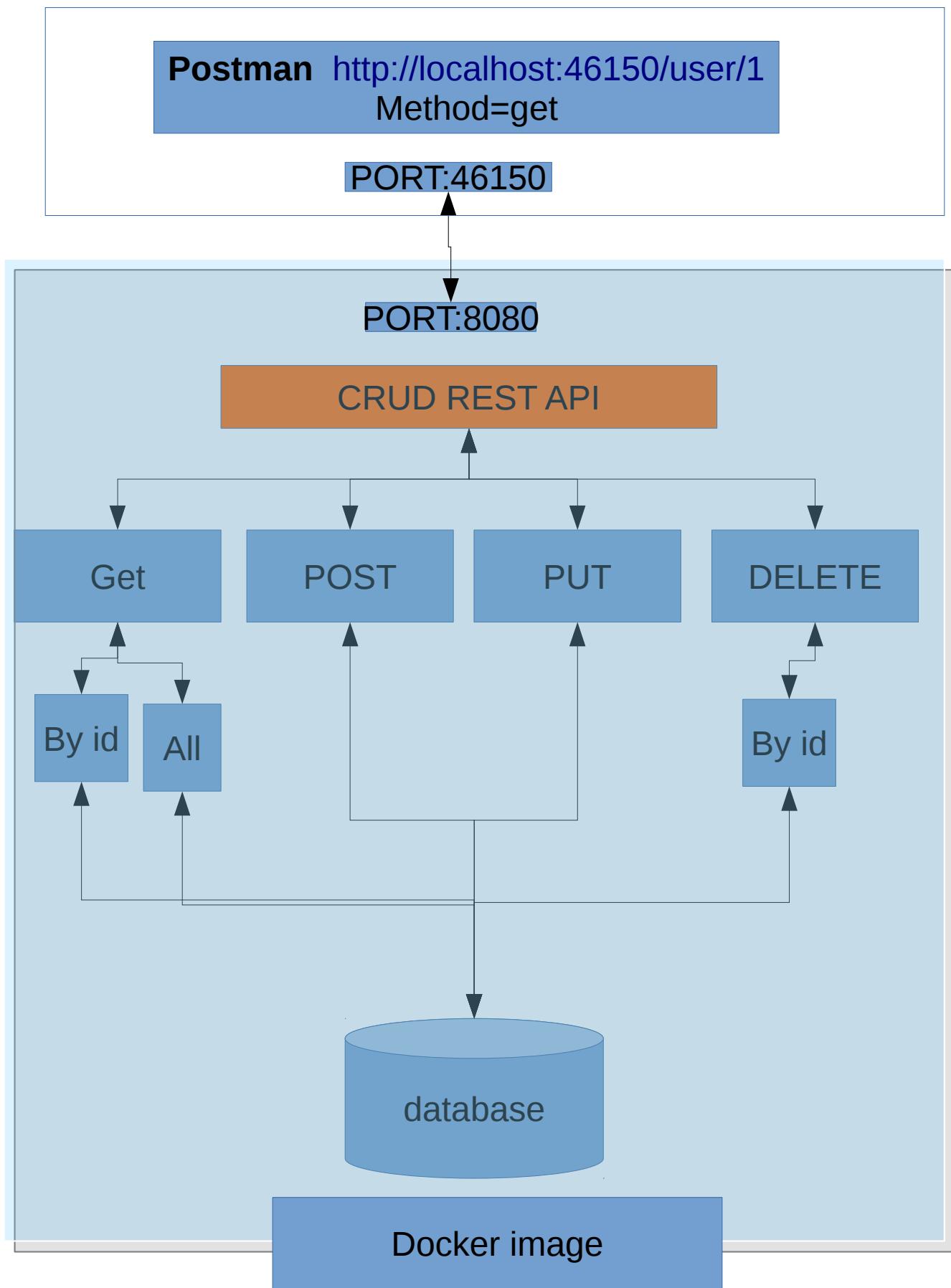
A screenshot of a Linux desktop environment. On the left, there's a dock with icons for various applications like a terminal, file manager, browser, and code editor. The main area shows two terminal windows. The first window is titled "docker exec -it 23aad069902a /bin/bash" and shows the command "ls -l" being run, displaying a directory listing with files like mysql-apt-config\_0.8.13-1\_all.deb, node\_modules, package-lock.json, package.json, and service.js. The second window is titled "vishal@vishal-inspiron-15-3567: ~/Vishal/Sem6-work/..." and shows the command "ls" being run, displaying a directory listing with files like .gitignore, Dockerfile, README.md, and requirements.txt.

```
root@23aad069902a:~/crud# ls -l
total 68
-rw-r--r-- 1 root root 35560 Apr 29 2019 mysql-apt-config_0.8.13-1_all.deb
drwxr-xr-x 61 root root 4096 Apr 26 11:29 node_modules
-rw-r--r-- 1 root root 16802 Apr 26 10:45 package-lock.json
-rw-r--r-- 1 root root 345 Apr 26 11:28 package.json
-rw-r--r-- 1 root root 2818 Apr 26 11:43 service.js
root@23aad069902a:~/crud# _
```

### Api Documentation:

Method	URL	Description
GET by id	<a href="http://localhost:46150/user/:id">http://localhost:46150/user/:id</a>	Get user by user id
Get (all)	<a href="http://localhost:46150/user">http://localhost:46150/user</a>	Get all users
POST	<a href="http://localhost:46150/user">http://localhost:46150/user</a>	Add user in database
PUT	<a href="http://localhost:46150/user">http://localhost:46150/user</a>	Update username using userid
DELETE(id)	<a href="http://localhost:46150/user/id">http://localhost:46150/user/id</a>	Delete user using userid

## Flow diagram



## **dockerfile:**

```
FROM crud2:latest

RUN cd
RUN cd /root/crud
RUN /etc/init.d/mysql stop

RUN mysqld_safe --skip-grant-tables &

EXPOSE 8080

CMD [ "node", "index.js" ]
```

---

## **service.js:**

```
// Including the file and initialize
const express = require('express');
const app = express();
const bodyParser = require('body-parser');
const mysql = require('mysql');

app.use(bodyParser.json());
app.use(bodyParser.urlencoded({
    extended : true
}));

const mysql_conn = mysql.createConnection({
    host: 'localhost',
    user: 'vishal',
    database: 'userdb'
});
mysql_conn.connect();

app.get('/',function (req,res){
```

```
        return res.send({error:true, message:
"Rest service using CRUD ops."} );
} );

app.get('/user', function (req, res) {
    mysql_conn.query('SELECT * FROM user',
function (error, results, fields) {
        if (error) throw error;
        res.end(JSON.stringify(results));
    });
} );

// Retrieve user with id
app.get('/user/:id', function (req, res) {

    let user_id = req.params.id;

    if (!user_id) {
        return res.status(400).send({ error: true,
message: 'Enter userID' });
    }

    mysql_conn.query('SELECT * FROM tasks where
id=?', user_id, function (error, results, fields)
{
        if (error) throw error;
        return res.end(JSON.stringify(results));
    });
} );

// Add a new user
app.post('/user', function (req, res) {
    var userdata=req.body;
//    console.log(userdata);
    if (!req.body.id) {
```

```
        return res.status(400).send({ error:true,
message: 'Please enter userid' });
    }

    mysql_conn.query('INSERT INTO user
SET ?',userdata, function (error, results, fields)
{
    if (error) throw error;
    else{
        console.log("User added
successfully");
        res.end(JSON.stringify(results));
    }
});
});

// Update username with id
app.put('/user', function (req, res) {

    let user_id = req.body.id;
    let name = req.body.name;
//    console.log('my user id '+user_id+name);

    if (!user_id || !name) {
        return res.status(400).send({ error: task,
message: 'Enter user id and name' });
    }

    mysql_conn.query("UPDATE user SET name = ?
WHERE id = ?", [name, user_id], function (error,
results, fields) {
        if (error) throw error;
        return res.send({ error: false, data:
results, message: 'User updated successfully.' });
    });
});
```

```
// Delete user
app.delete('/user/:id', function (req, res) {

    let user_id = req.params.id;

    mysql_conn.query('DELETE FROM user WHERE id
= ?', [user_id], function (error, results, fields)
{
    if (error) throw error;
    return res.send({ error: false, data:
results, message: 'User deleted successfully.' });
});
});

// All other requests redirect to 404
app.all('*', function (req, res, next) {
    return res.send('Page not found');
next();
});

//Port must be set to 8080 because incoming http
request are routed from 80 to 8080

app.listen(8080, function(){
    console.log('App is running on port
8080');
});
```

---

## **package.json**

```
{  
  "name": "express-node-rest-project",  
  "version": "1.0.0",  
  "description": "",  
  "main": "service.js",  
  "scripts": {  
    "test": "echo \\\"Error: no test  
specified\\\" && exit 1"  
  },  
  "keywords": [ ],  
  "author": "",  
  "license": "ISC",  
  "dependencies": {  
    "body-parser": "^1.19.0",  
    "express": "^4.17.1",  
    "mysql": "^2.16.0"  
  }  
}
```

---

## Snapshot:

### 1) Docker images && container running

vishal@vishal-inspiron-15-3567: ~/Vishal/Sem6-work/pucsd2020-Tasks/Training/project2

```
+ project2 git:(master) X docker images
REPOSITORY          TAG      IMAGE ID      CREATED       SIZE
crudrest            dockerfile  b58dbc12f077  14 minutes ago  1.14GB
crud2               latest    b4928ea4ac53  18 minutes ago  1.14GB
vishal              latest    33ceed915f0c  2 hours ago   1.1GB
helloworld          latest    9dd3633380cc  2 hours ago   1.1GB
dockerbuild         dockerfile  53caab90b262  10 hours ago  1.1GB
new2                latest    a515a462ddd  10 hours ago  1.1GB
mysql-node-ubuntu   latest    az1d9eb750d8  11 hours ago  1.1GB
mysql_ubuntu        dockerfile  f9b78f8c5341  12 hours ago  644MB
crud-service        latest    e9a0cfb89b3d  30 hours ago  1GB
ubuntu              latest    1d622ef86b13  3 days ago   73.9MB
debian              latest    971452c94376  2 months ago  114MB
herreraluis/mysql-ubuntu  latest    293d44326d32  3 years ago   345MB
+ project2 git:(master) X _
```

```
+ project2 git:(master) X docker ps
CONTAINER ID        IMAGE             COMMAND           CREATED          STATUS            PORTS          NAMES
23aad069902a        b58dbc12f077   "/bin/bash"       13 minutes ago   Up 13 minutes   0.0.0.0:46150->8080/tcp   crazy_goldwasser
```

The screenshot shows a terminal window with two tabs. The left tab displays a list of Docker images with their repository names, tags, image IDs, creation times, and sizes. The right tab shows a single container named 'crazy\_goldwasser' running on port 8080.

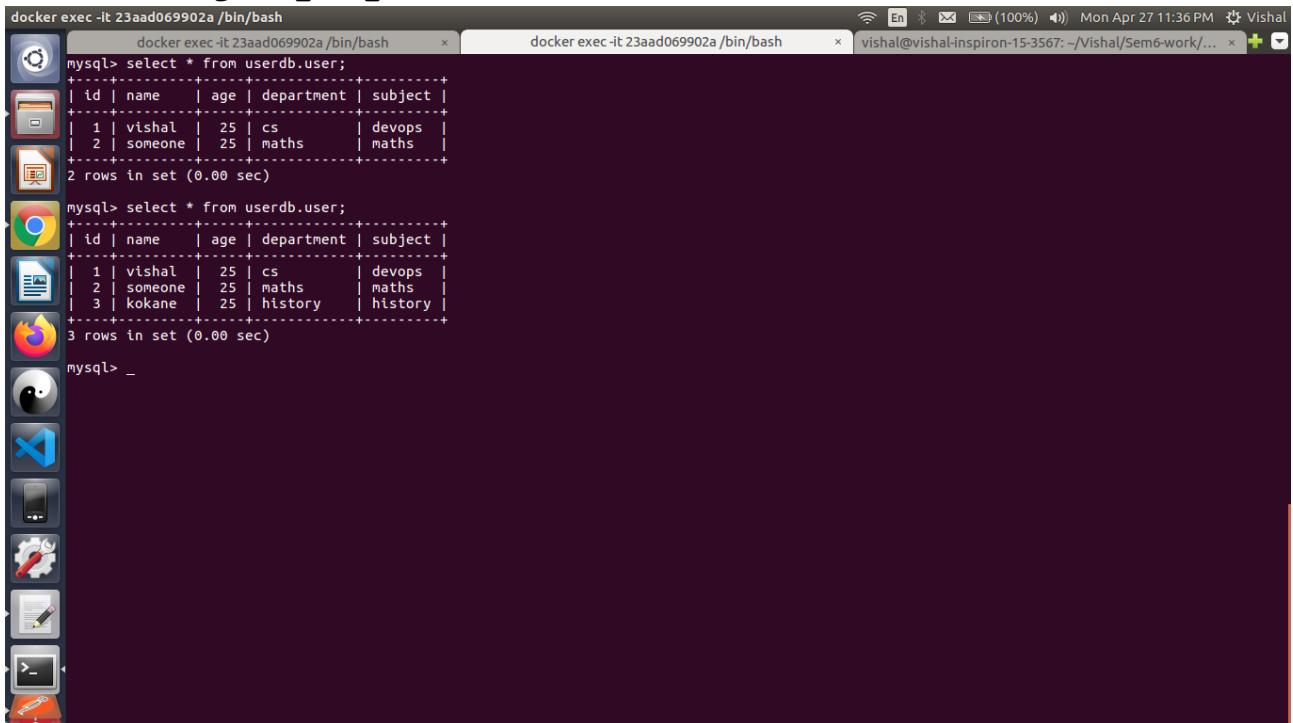
### 2) Nodejs service running on port 8080

vishal@vishal-inspiron-15-3567: ~/Vishal/Sem6-work/pucsd2020-Tasks/Training/project2

```
root@23aad069902a:~/crud# node service.js
App is running on port 8080
User added successfully
```

The screenshot shows a terminal window with three tabs. The middle tab contains the command 'node service.js' and its output, indicating that the application is running on port 8080 and a user has been added successfully.

### 3) Another same container running for checking mysql data

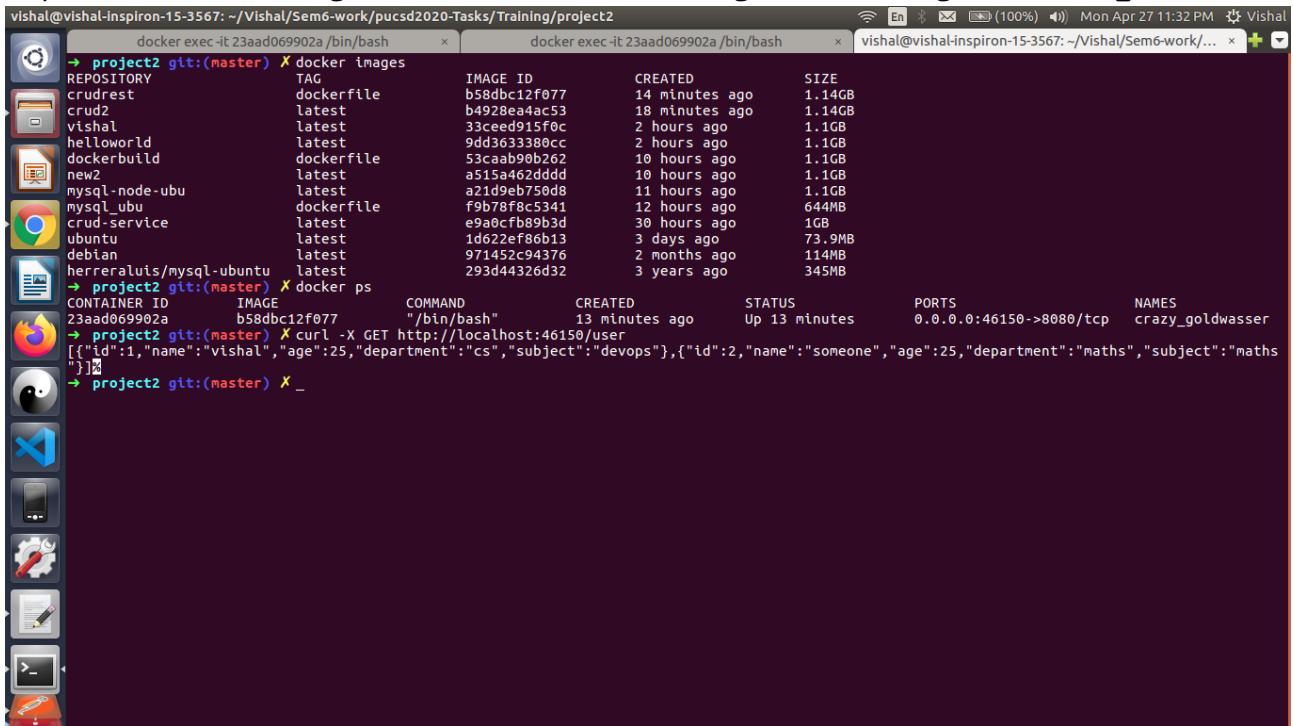


```
mysql> select * from userdb.user;
+----+-----+-----+-----+-----+
| id | name | age | department | subject |
+----+-----+-----+-----+-----+
| 1  | vishal | 25 | cs       | devops   |
| 2  | someone | 25 | maths    | maths    |
| 3  | kokane  | 25 | history  | history  |
+----+-----+-----+-----+-----+
2 rows in set (0.00 sec)

mysql> select * from userdb.user;
+----+-----+-----+-----+-----+
| id | name | age | department | subject |
+----+-----+-----+-----+-----+
| 1  | vishal | 25 | cs       | devops   |
| 2  | someone | 25 | maths    | maths    |
| 3  | kokane  | 25 | history  | history  |
+----+-----+-----+-----+-----+
3 rows in set (0.00 sec)

mysql> _
```

### 4) Validating service using CURL get request



```
vishal@vishal-inspiron-15-3567: ~/Vishal/Sem6-work/pucsd2020-Tasks/Training/project2
→ project2 git:(master) X docker images
REPOSITORY          TAG      IMAGE ID      CREATED        SIZE
crudrest            dockerfile  b58dbc12f077  14 minutes ago  1.14GB
crud2               latest    b4928ea4ac53  18 minutes ago  1.14GB
vishal              latest    33ceed915f0c  2 hours ago   1.1GB
helloworld          latest    9dd3633380cc  2 hours ago   1.1GB
dockerbuild         dockerfile  53caab99b262  10 hours ago  1.1GB
new2                latest    a515a462ddd  10 hours ago  1.1GB
mysql-node-ubuntu   latest    a71d9eb750d8  11 hours ago  1.1GB
mysql_ubuntu         dockerfile  f9b78f8c5341  12 hours ago  644MB
crud-service         latest    e9a0cfb89b3d  30 hours ago  1GB
ubuntu              latest    1d622ef86b13  3 days ago   73.9MB
debian              latest    971452c94376  2 months ago  114MB
herreraluis/mysql-ubuntu latest    293d44326d32  3 years ago   345MB

→ project2 git:(master) X docker ps
CONTAINER ID        IMAGE           COMMAND       CREATED        STATUS          PORTS          NAMES
23aad069902a        b58dbc12f077   "/bin/bash"   13 minutes ago  Up 13 minutes   0.0.0.0:46150->8080/tcp   crazy_goldwasser
→ project2 git:(master) X curl -X GET http://localhost:46150/user
[{"id":1,"name":"vishal","age":25,"department":"cs","subject":"devops"}, {"id":2,"name":"someone","age":25,"department":"maths","subject":"maths"}]
→ project2 git:(master) X _
```

## 5) Postman POST method:

The screenshot shows the Postman application interface. In the top navigation bar, there are buttons for NEW, Runner, Import, and a search bar. To the right, it displays the date and time (Mon Apr 27 11:35 PM), the user's name (Vishal), and various system icons. The main workspace has a yellow header bar with the URL 'localhost:46150/user/' and a status message: 'The latest version of Postman is out with huge improvements! Update the app for a better experience. Download'. Below this, there are tabs for History, Collections, and a 'Clear all' button. The main area shows a POST request to 'localhost:46150/user/'. The 'Body' tab is selected, showing a raw JSON payload: 

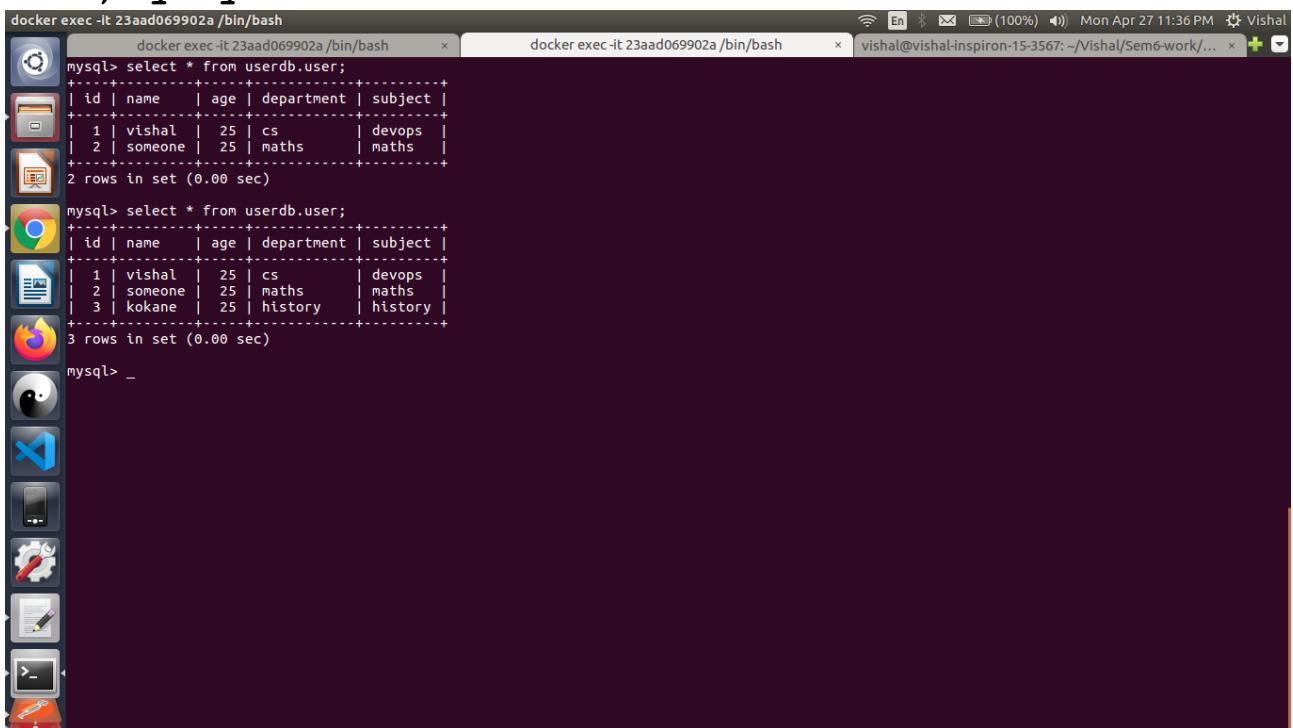
```
{"id":3,"name":"kokane","age":25,"department":"history","subject":"history"}
```

. Other tabs include Authorization, Headers (1), Pre-request Script, Tests, Body, Cookies, Headers (4), and Test Results. The status bar at the bottom indicates a 200 OK response with a time of 94 ms.

### 5.1) User added Successfully

The screenshot shows a terminal window with three tabs. The left tab shows the command 'docker exec -it 23aad069902a /bin/bash'. The middle tab shows the command 'docker exec -it 23aad069902a /bin/bash' and the output: 'root@23aad069902a:~/crud# node service.js' followed by 'App is running on port 8080'. The right tab shows the command 'vishal@vishal-inspiron-15-3567: ~/Vishal/Sem6-work...' and the output: 'User added successfully'. The terminal has a dark theme with a vertical icon bar on the left.

## 5.2) Mysql result of POST

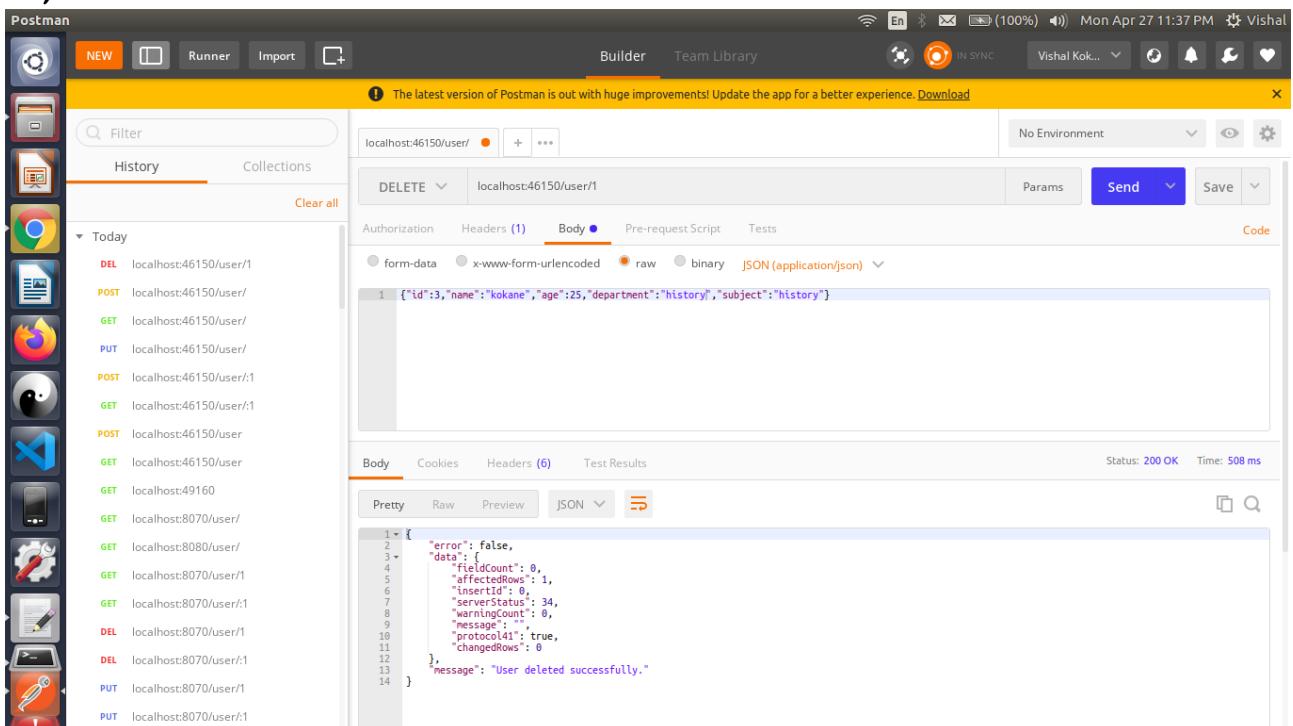


```
docker exec -it 23aad069902a /bin/bash
[docker exec -it 23aad069902a /bin/bash] docker exec -it 23aad069902a /bin/bash [vishal@vishal-inspiron-15-3567: ~/Vishal/Sem6-work/...]
mysql> select * from userdb.user;
+----+-----+-----+-----+-----+
| id | name | age | department | subject |
+----+-----+-----+-----+-----+
| 1 | vishal | 25 | cs | devops |
| 2 | someone | 25 | maths | maths |
+----+-----+-----+-----+-----+
2 rows in set (0.00 sec)

mysql> select * from userdb.user;
+----+-----+-----+-----+-----+
| id | name | age | department | subject |
+----+-----+-----+-----+-----+
| 1 | vishal | 25 | cs | devops |
| 2 | someone | 25 | maths | maths |
| 3 | kokane | 25 | history | history |
+----+-----+-----+-----+-----+
3 rows in set (0.00 sec)

mysql> _
```

## 6) Postman delete data with user id =1



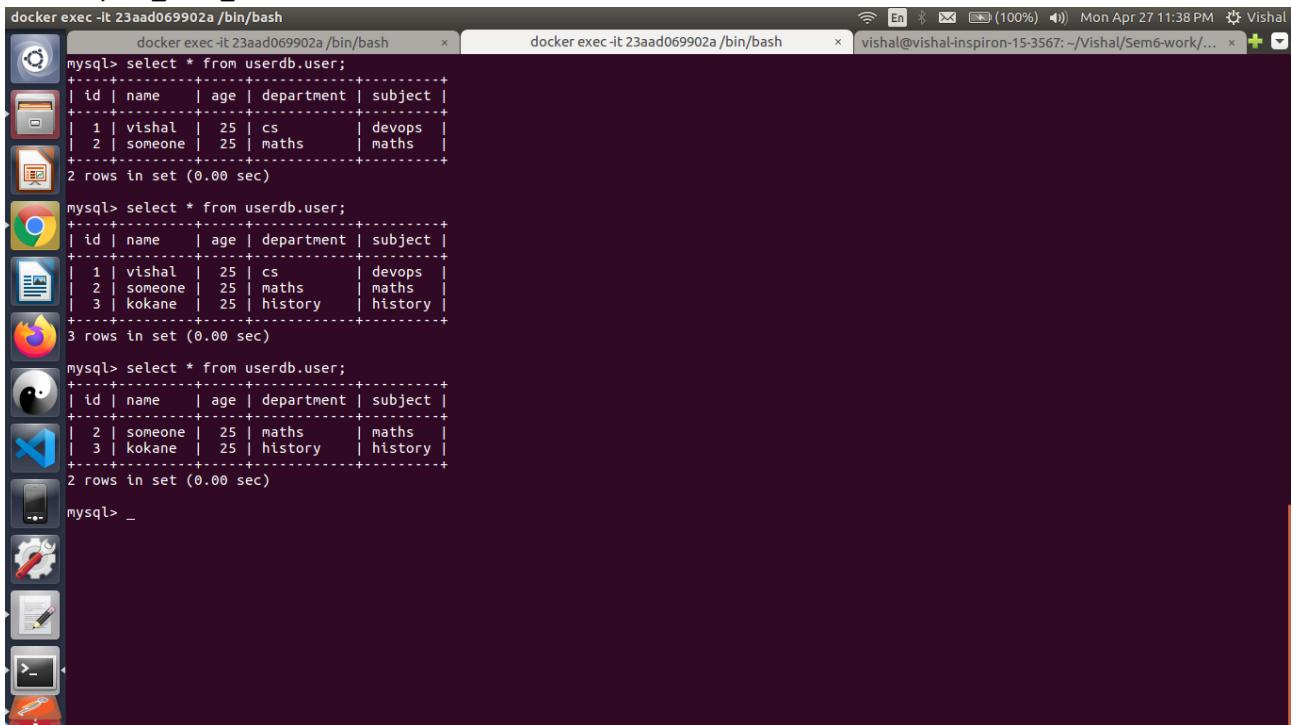
The screenshot shows the Postman application interface. On the left, there's a sidebar with various icons and a list of recent requests. The main area has a yellow header bar with a message about the latest version. Below it, there's a search bar and tabs for 'History' and 'Collections'. The main workspace shows a 'DELETE' request to 'localhost:46150/user/1'. The 'Body' tab is selected, showing a JSON payload: 

```
{"id":3,"name":"kokane","age":25,"department":"history","subject":"history"}
```

. The 'Params' tab is empty. To the right, there are 'Send' and 'Save' buttons. At the bottom, the response is shown with a status of '200 OK' and a time of '508 ms'. The response body is a JSON object:

```
1  {
2      "error": false,
3      "data": {
4          "fieldCount": 0,
5          "affectedRows": 1,
6          "insertId": 0,
7          "serverStatus": 34,
8          "warningCount": 0,
9          "message": "",
10         "protocol41": true,
11         "changedRows": 0
12     },
13     "message": "User deleted successfully."
14 }
```

## 6.2) Mysql status



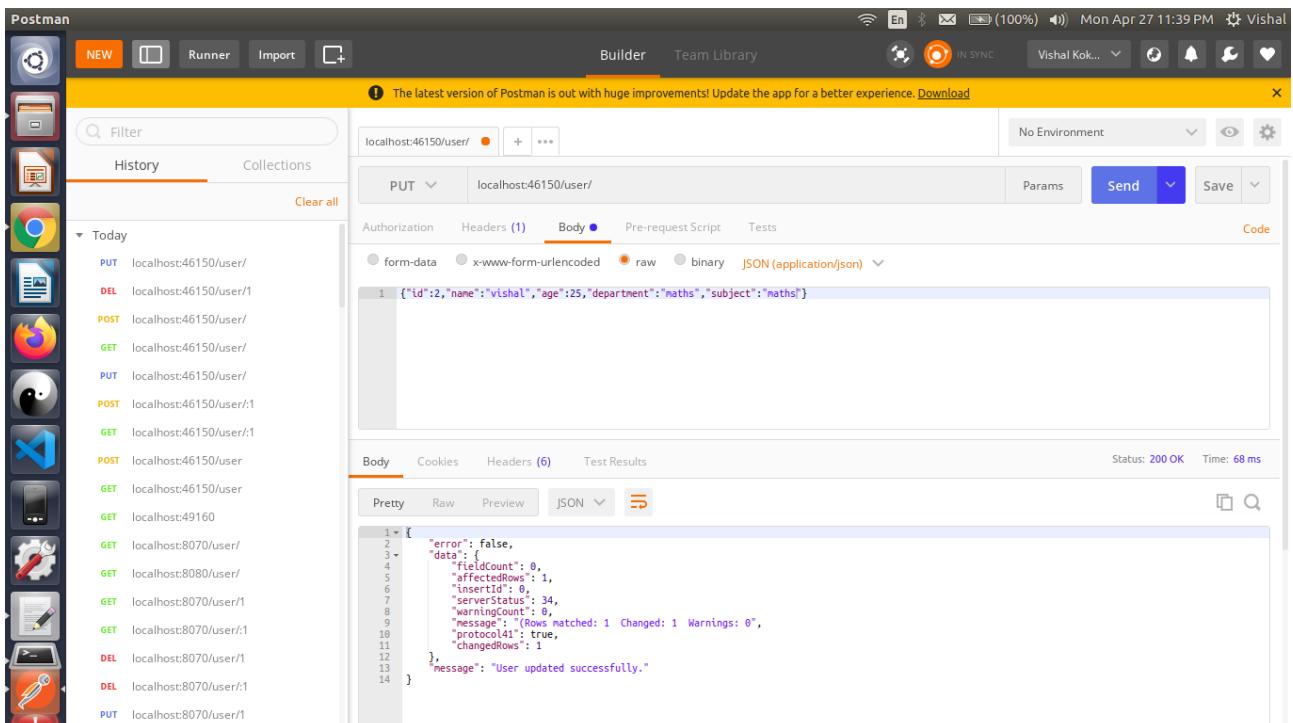
```
docker exec -it 23aad069902a /bin/bash
docker exec -it 23aad069902a /bin/bash
vishal@vishal-inspiron-15-3567: ~/Vishal/Sem6-work/... + docker exec -it 23aad069902a /bin/bash
mysql> select * from userdb.user;
+----+-----+-----+-----+
| id | name | age | department | subject |
+----+-----+-----+-----+
| 1 | vishal | 25 | cs | devops |
| 2 | someone | 25 | maths | maths |
+----+-----+-----+-----+
2 rows in set (0.00 sec)

mysql> select * from userdb.user;
+----+-----+-----+-----+
| id | name | age | department | subject |
+----+-----+-----+-----+
| 1 | vishal | 25 | cs | devops |
| 2 | someone | 25 | maths | maths |
| 3 | kokane | 25 | history | history |
+----+-----+-----+-----+
3 rows in set (0.00 sec)

mysql> select * from userdb.user;
+----+-----+-----+-----+
| id | name | age | department | subject |
+----+-----+-----+-----+
| 2 | someone | 25 | maths | maths |
| 3 | kokane | 25 | history | history |
+----+-----+-----+-----+
2 rows in set (0.00 sec)

mysql> _
```

## 7) Postman PUT :update user name using userid



The screenshot shows the Postman application interface. On the left, there's a sidebar with various icons and a list of recent requests. The main area has a header bar with the URL "localhost:46150/user/" and several tabs: "Builder", "Team Library", "Vishal Kok...", and "IN SYNC". Below the header, there are sections for "PUT" method, "localhost:46150/user/" path, "Params", "Send", and "Save". Under the "Body" tab, the "raw" option is selected, and the JSON payload is set to:

```
{"id":2,"name":"vishal","age":25,"department":"maths","subject":"maths"}
```

At the bottom, the response status is shown as "Status: 200 OK Time: 68 ms". The response body is displayed in a JSONpretty-printed format:

```
1: [
  2:   {
    3:     "error": false,
    4:     "data": {
      5:       "fieldCount": 0,
      6:       "affectedRows": 1,
      7:       "insertId": 0,
      8:       "serverStatus": 34,
      9:       "warningCount": 0,
     10:       "message": "(Rows matched: 1 Changed: 1 Warnings: 0",
     11:       "protocol41": true,
     12:       "changedRows": 1
     13:     },
     14:     "message": "User updated successfully."
  }
]
```

## 7.2) Mysql update status

```
docker exec -it 23aad069902a /bin/bash
mysql> select * from userdb.user;
+----+-----+-----+-----+-----+
| id | name | age | department | subject |
+----+-----+-----+-----+-----+
| 1 | vishal | 25 | cs | devops |
| 2 | someone | 25 | maths | maths |
+----+-----+-----+-----+-----+
2 rows in set (0.00 sec)

mysql> select * from userdb.user;
+----+-----+-----+-----+-----+
| id | name | age | department | subject |
+----+-----+-----+-----+-----+
| 1 | vishal | 25 | cs | devops |
| 2 | someone | 25 | maths | maths |
| 3 | kokane | 25 | history | history |
+----+-----+-----+-----+-----+
3 rows in set (0.00 sec)

mysql> select * from userdb.user;
+----+-----+-----+-----+-----+
| id | name | age | department | subject |
+----+-----+-----+-----+-----+
| 2 | someone | 25 | maths | maths |
| 3 | kokane | 25 | history | history |
+----+-----+-----+-----+-----+
2 rows in set (0.00 sec)

mysql> select * from userdb.user;
+----+-----+-----+-----+-----+
| id | name | age | department | subject |
+----+-----+-----+-----+-----+
| 2 | vishal | 25 | maths | maths |
| 3 | kokane | 25 | history | history |
+----+-----+-----+-----+-----+
2 rows in set (0.00 sec)

mysql> _
```

## 8) Postman get all records

The latest version of Postman is out with huge improvements! Update the app for a better experience. [Download](#)

localhost:46150/user/

GET localhost:46150/user/

Headers (1)

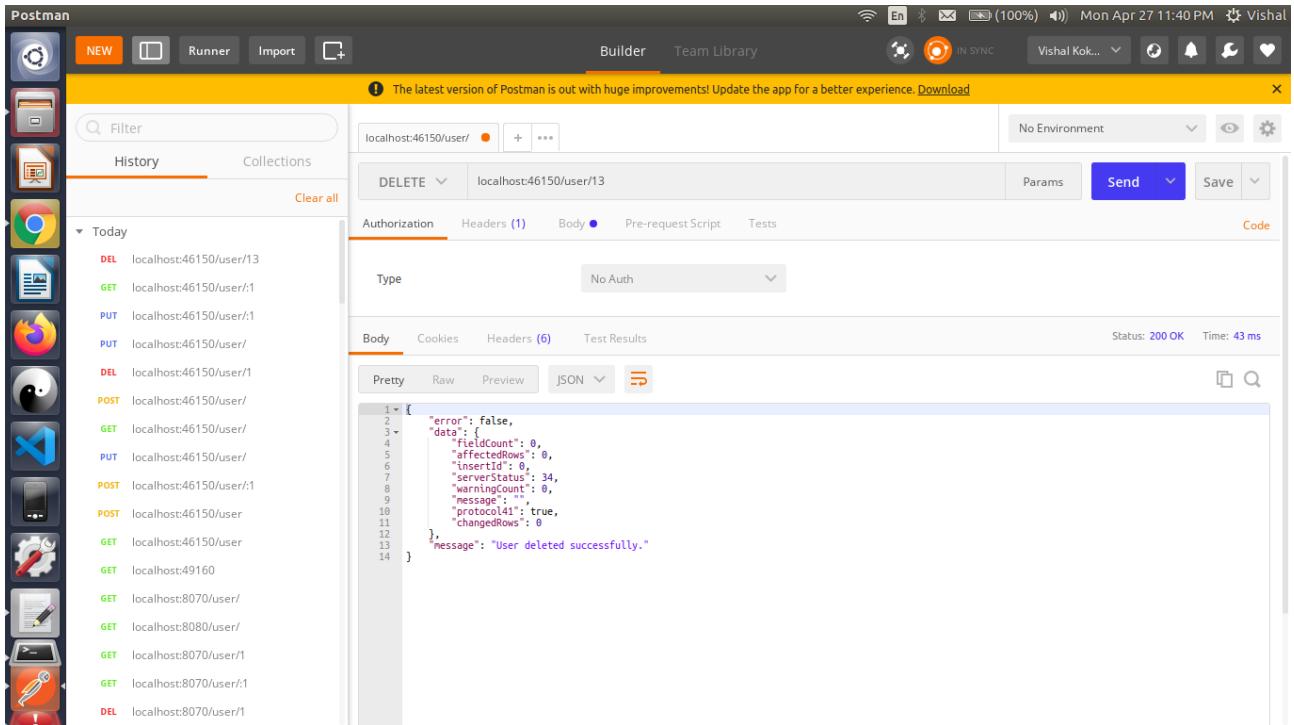
Type: No Auth

Status: 200 OK Time: 37 ms

Pretty Raw Preview Text

```
[{"id":1,"name":"vishal","age":25,"department":"cs","subject":"devops"}, {"id":2,"name":"someone","age":25,"department":"maths","subject":"maths"}]
```

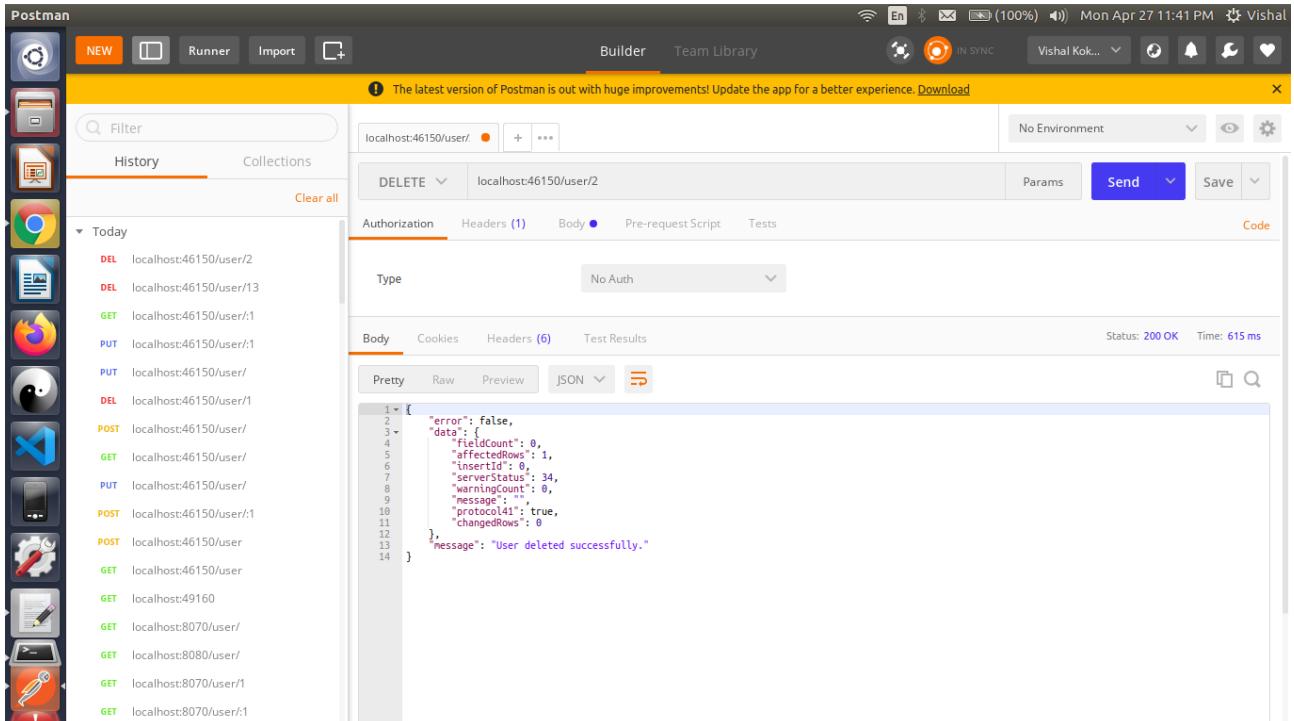
## 9) Postman delete fail record not exist



The screenshot shows the Postman application interface. In the top right corner, it says "Mon Apr 27 11:40 PM" and "Vishal". The main area displays a DELETE request to "localhost:46150/user/13". The response status is "200 OK" and the time is "43 ms". The response body is a JSON object:

```
1 < [  
2   "error": false,  
3   "data": {  
4     "fieldCount": 0,  
5     "affectedRows": 0,  
6     "insertId": 0,  
7     "serverStatus": 34,  
8     "warningCount": 0,  
9     "message": "",  
10    "protocol41": true,  
11    "changedRows": 0  
12  },  
13  "message": "User deleted successfully."  
14 ]
```

## 10) Postman delete Success



The screenshot shows the Postman application interface. In the top right corner, it says "Mon Apr 27 11:41 PM" and "Vishal". The main area displays a DELETE request to "localhost:46150/user/2". The response status is "200 OK" and the time is "615 ms". The response body is a JSON object:

```
1 < [  
2   "error": false,  
3   "data": {  
4     "fieldCount": 0,  
5     "affectedRows": 1,  
6     "insertId": 0,  
7     "serverStatus": 34,  
8     "warningCount": 0,  
9     "message": "",  
10    "protocol41": true,  
11    "changedRows": 0  
12  },  
13  "message": "User deleted successfully."  
14 ]
```

## 11) Postman Get by id

The screenshot shows the Postman application interface. The left sidebar displays a history of requests, including various GET, POST, PUT, and DELETE requests to localhost:46150 and localhost:8070. The main workspace shows a GET request to `localhost:46150/user/:3`. The response status is `200 OK` with a time of `72 ms`. The response body is a JSON array containing one element: `[{"id":3,"name":"kokane","age":25,"department":"history","subject":"history"}]`.